Warmup 1: Potentiometer

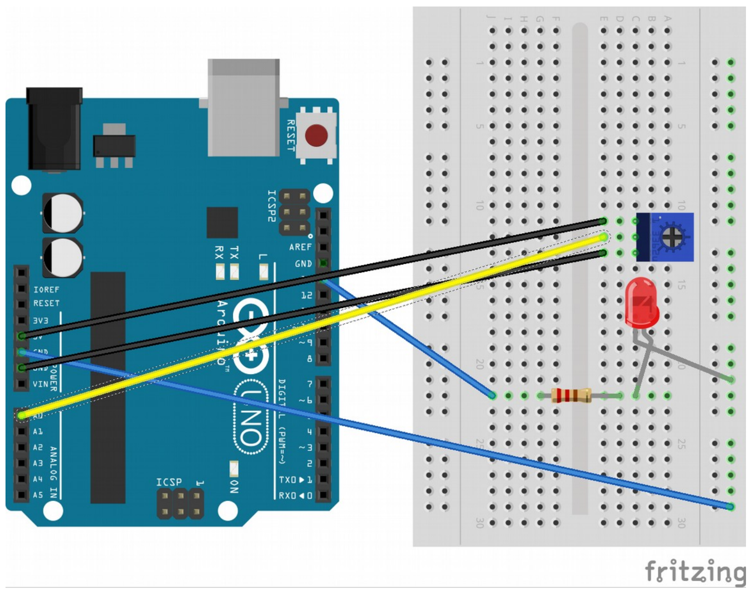
Description:

This was a light dimmer containing a trim pot, a 220Ω resistor, a red led and five cords, one from ground to the ground area of the half breadboard, one from digital pin 13 to A22 on the breadboard, one from ground to the right side of the trim pot, one from A0 to the middle pin of the trim pot, and one from the 5V digital pint to the left side of the trim pot. The potentiometer would dim the light of the led according to where I turned the potentiometer.

Issues:

There weren't really any issues with this build. My biggest issues were figuring out how to connect the trim pot correctly so that my circuit worked. I also ended up switching my code from the one at https://www.arduino.cc/en/Tutorial/Potentiometer to a different implementation that worked better. I think what I learned on this build was to be more cautious of the code that I find because although it may technically work, there could be a better implementation.

Diagram of Build



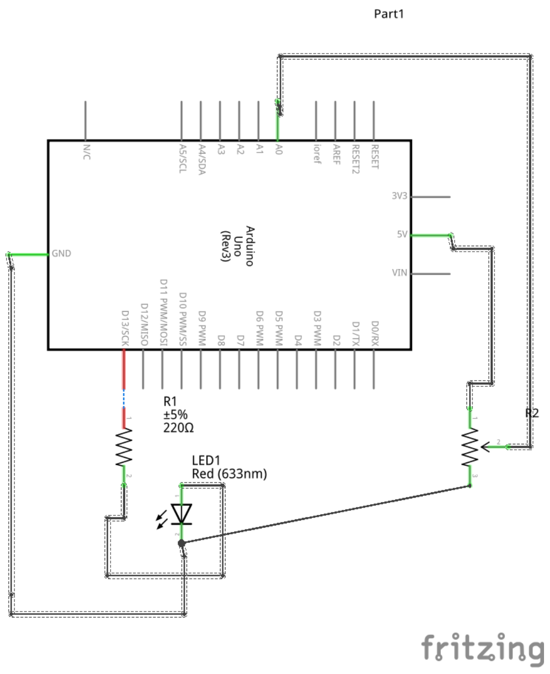
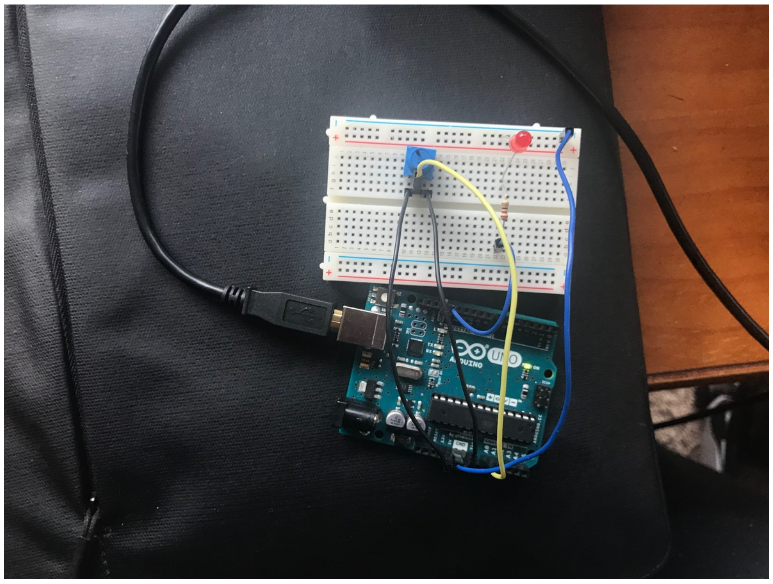


Photo of Build



Code for Build

```
/*         Edited          from          code          found          on
https://www.arduino.cc/en/Tutorial/AnalogInOutSerial */

const int ledPin = 13; // The pin connected to the led.
const int analogPin = A0; //Input reader for potentionmeter

int potentiometerVal = 0; //value of the potentiometer in the moment.
int changeVal = 0; //value we are going to set the LED at.

void setup() {
}

void loop() {
  potentiometerVal = analogRead(analogPin); //read the potentiometer
  //map the range of the potentiometer to the range of the led.
  changeVal = map(potentiometerVal, 0, 1023, 0, 255);
  analogWrite(ledPin, changeVal); //change the value of the led.
  delay(2);

}
```

Example of Output

Warmup 2: Door Bell

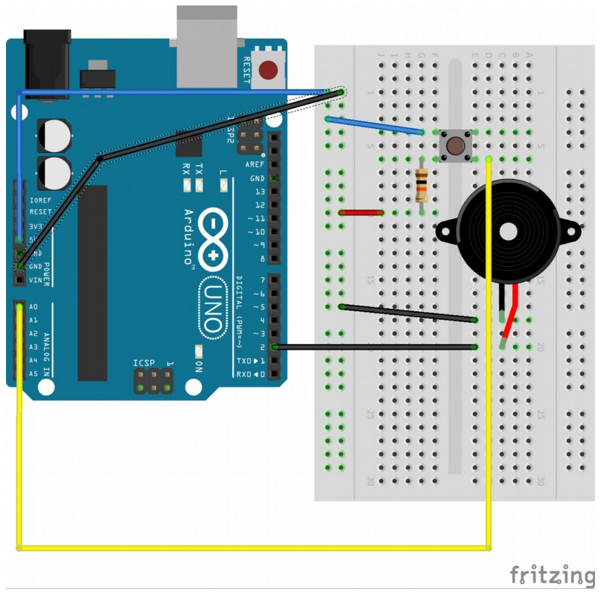
Description:

This was an piezo buzzer build. The goal have a buzzer that would sound a tone when we pressed a button not do anything when we didn't press anything. This build contained a button, seven wires, a 10KΩ resistor and a piezo buzzer. The wires were connected between digital pin 2 and one leg of the buzzer, one leg of the buzzer and ground, ground to the breadboard, the 5V digital pin to the breadboard, ground on the breadboard to the resistor, and a wire going from A0 to the button.

Issues:

I didn't really have any issues with this build once I realized I had to use tone instead of using analogWrite or digitalWrite to make a sound. I was impressed by how notes you could use with the buzzer and I was surprised at how hard it was to get the tone correct so I could get the beginning of Prelude in C playing. I went through multiple code versions to get it to sound kinda okay.

Diagram of Build



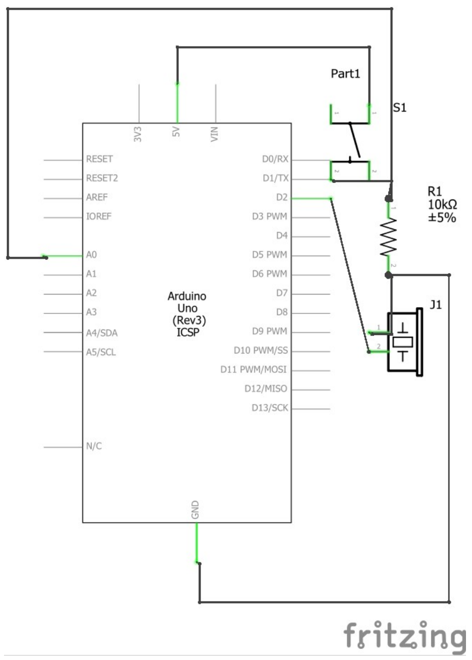
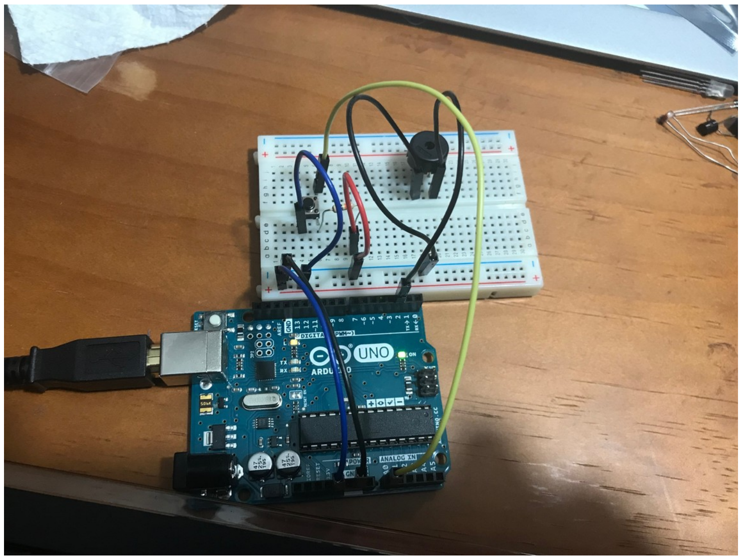


Photo of Build



Code for Build

```
#include <Button.h>

Button button = Button(A0, PULLDOWN);
//const int buttonPin = A0;
const int ringerPin = 2;

void setup() {
  // put your setup code here, to run once:

}

void loop() {
  if(button.isPressed()) {
    if(!button.stateChanged()){
      tone(ringerPin, 208, 5000);
      delay(300);
      tone(ringerPin, 523, 5000);
      delay(300);
      tone(ringerPin, 659, 5000);
      delay(300);
      tone(ringerPin, 208, 5000);
      delay(300);
      tone(ringerPin, 523, 5000);
      delay(300);
      tone(ringerPin, 659, 5000);
      delay(300);
      tone(ringerPin, 208, 5000);
      delay(300);
      tone(ringerPin, 523, 5000);
      delay(300);
    }
    noTone(ringerPin);
  } else {
    noTone(ringerPin);
  }

}
```

Example of Output

Build 1: Doorbell with Audio/Visual Components

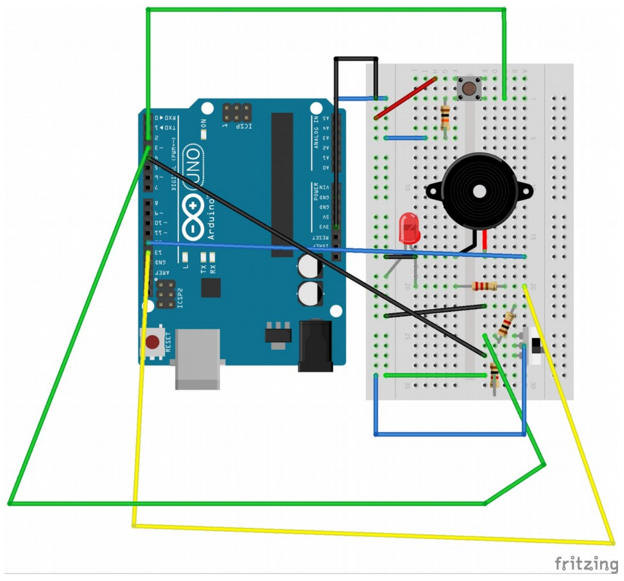
Description:

This was a build that set it up so that a slide switch controlled whether the buzzer would ring or whether the led would light up. This build contained a piezo buzzer, a red led, a slide switch, a 2 220Ω resistors, a button, and 11 wires.

Issues:

The biggest issue I had was trying to get the Arduino to read the input from the slide switch. I kept running into an issue where it wouldn't seem to read the switch at all. I thought it might be a software issue so I kept trying to edit the code on how the Arduino was reading from the switch. It turned out I'd forgotten the resistor to the button that controlled everything and that was why nothing was working correctly. I was surprised at how much that small piece affected my circuit.

Diagram of Build



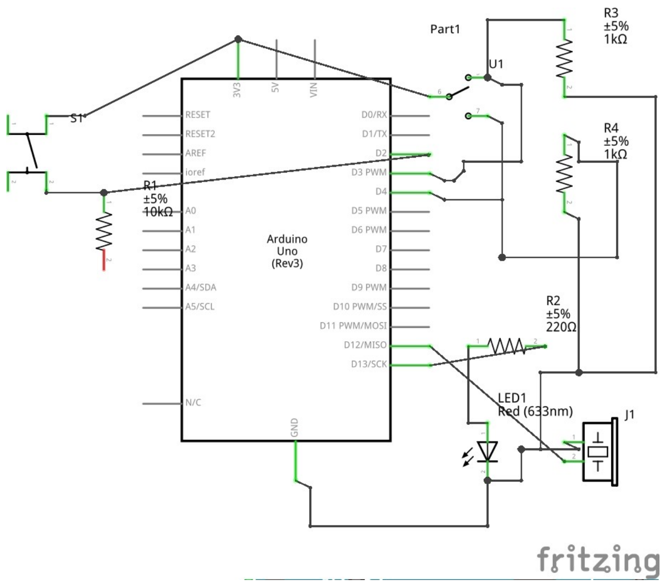
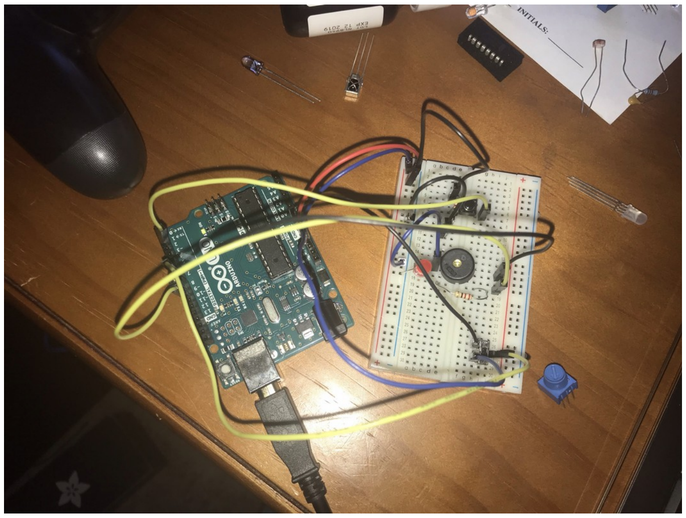


Photo of Build



Code for Build

```
/* Changed the code https://www.arduino.cc/en/Tutorial/Button
Tried to read the input from a slide switch. */

//represents button
const int buttonPin = 2;

//represents input from the left side of switch
const int leftPin = 3;

//represents input from right side of switch
const int rightPin = 4;

//represents output to buzzer
const int tonePin = 12;

// represents output to led
const int ledPin = 13;

int buttonState = 0; // variable for reading the pushbutton status
int rightState = 0; //variable for reading the right side of the
switch
int leftState = 0; //variable for reading the left side of the
switch.


void setup() {
 // initialize the LED and tone pin as an output:
 pinMode(ledPin, OUTPUT);
 pinMode(tonePin, OUTPUT);

 // initialize the pushbutton and slide switch pin as an input:
```

```
 pinMode(buttonPin, INPUT);
 pinMode(rightPin, INPUT);
 pinMode(leftPin, INPUT);
}

void loop() {
 // read the state of the pushbutton value:
 buttonState = digitalRead(buttonPin);
 //read the state of the right side of the switch:
 rightState = digitalRead(rightPin);
 //read the state of the left side of the switch:
 leftState = digitalRead(leftPin);

 // check if the pushbutton is pressed. If it is, the buttonState is
HIGH:
  if (buttonState == HIGH) {
   //turn LED on:
      if(rightState == HIGH) {
        digitalWrite(ledPin, HIGH);
        noTone(tonePin);
      }

      if (leftState == HIGH) {
        digitalWrite(ledPin, LOW);
        tone(tonePin, 261);
      }
   } else {
      // turn everything off:
    digitalWrite(ledPin, LOW);
    noTone(tonePin);
   }
 }
```

Example of Output

Build 2: Use Shift Register + IR Remote to Drive Multiple Actuators

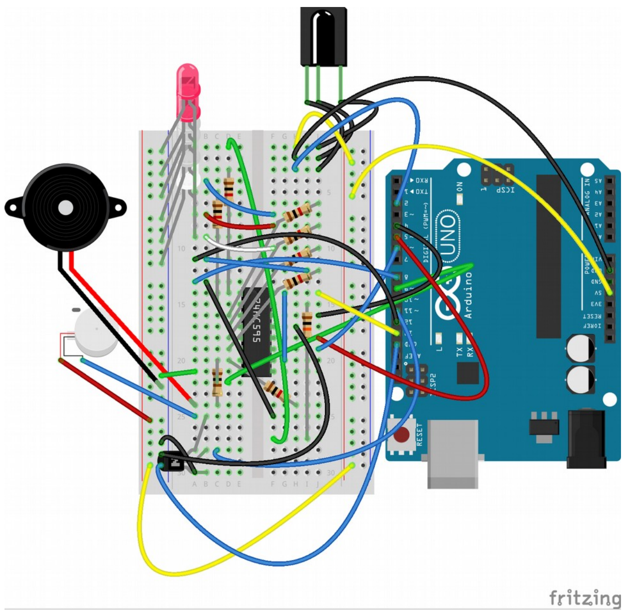
Description:

For this project, I decided to use 6 leds, the piezo buzzer, and the small vibration motor to create this build. The goal of the build was to use a shift register to connect 8 output devices to the IR receiver so we could have different output devices activate based on the input from the IR remote we used.

Issues:

So the first issue I ran into was that all my buttons weren't being recognized as individual buttons. I'd occasionally get errors that read the buttons as two different inputs at the same time or would read the buttons as all the buttons at once (that is what happened with my on button). So I had to shift from having individual buttons for each output device and bundle the output devices in some way. At most, I could use four buttons and the fourth button always ever didn't read as anything or came up as two different things. So I had to accept that in order for my build to work. Then I ran into an issue where my buzzer worked only one time and then for every test after that, the vibration motor would move but the buzzer wouldn't make a sound. I never figured out why the buzzer stopped working. That was really frustrating. I'm surprised in the amount of work that goes into reading a remote and how much you have to mess with how broadly you interpret the input to get anything to register as distinct inputs.

Diagram of Build



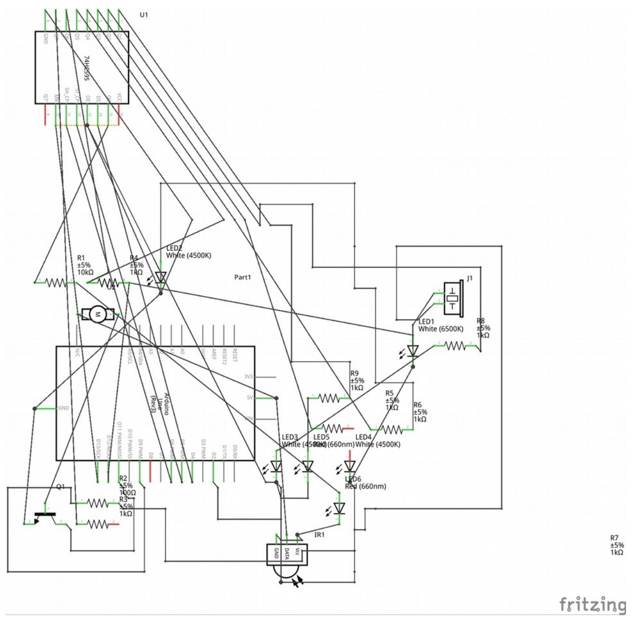
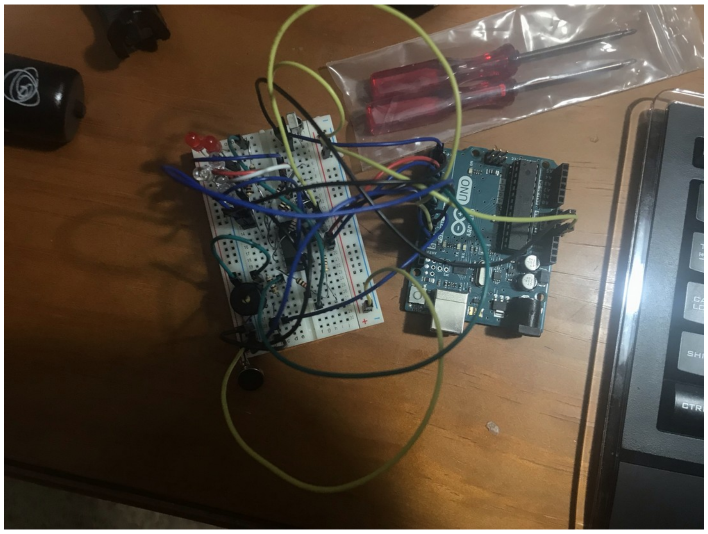


Photo of Build



Code for Build

```
#include "remoteinput.h"
/* Raw IR commander


                                                                       From
https://github.com/adafruit/IR-Commander/blob/master/ircommander.pde
combined with
    code  from  https://learn.adafruit.com/adafruit-arduino-lesson-4-
eight-leds/arduino-code
    and  https://www.precisionmicrodrives.com/content/how-to-drive-a-
vibration-motor-with-arduino-and-genuino/.
  Edited so it works with my remote and my set up.
*/

#define IRpin_PIN      PIND
#define IRpin          2

// the maximum pulse we'll listen for - 65 milliseconds is a long
time
#define MAXPULSE 65000
#define NUMPULSES 50

#define RESOLUTION 20

// What percent we will allow in variation to match the same code
#define FUZZINESS 55

// we will store up to 100 pulse pairs (this is -a lot-)
uint16_t pulses[NUMPULSES][2];  // pair is high and low pulse
uint8_t currentpulse = 0; // index for pulses we're storing

//Pins for the shift register
```

```
int registerPin = 13;
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;

byte leds = 0;

//Pins for controlling piezo buzzer
int buzzerReaderPin = 8;
int buzzerPin = 9;

//Pins for controlling motor
int motorPin = 11;
int motorReaderPin = 12;

//Used for interpreting motor and pin states;
int motorReaderState = 0;
int buzzerReaderState = 0;

void setup(void) {
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(registerPin, HIGH);
  Serial.begin(9600);
  Serial.println("Ready to decode IR!");
}

void loop(void) {
  leds = 0;
  int numberpulses;

  numberpulses = listenForIR();
  updateShiftRegister();

  Serial.print("Heard ");
  Serial.print(numberpulses);
  Serial.println("-pulse long IR signal");
              if       (IRcompare(numberpulses,       HeatDownSignal,
sizeof(HeatDownSignal) / 4)) {
    Serial.println("HEAT DOWN");
    bitSet(leds, 2);
    updateShiftRegister();

    bitSet(leds, 3);
    updateShiftRegister();
    delay(500);
  }
             if       (IRcompare(numberpulses,       RemoteColdSignal,
sizeof(RemoteColdSignal) / 4)) {
    Serial.println("COLD");
    bitSet(leds, 4);
    updateShiftRegister();

    bitSet(leds, 5);
    updateShiftRegister();
    delay(500);
  }
   if (IRcompare(numberpulses, HeatUpSignal, sizeof(HeatUpSignal) /
4)) {
    Serial.println("HEAT UP");
    bitSet(leds, 6);
    updateShiftRegister();

    bitSet(leds, 7);
    updateShiftRegister();
    delay(500);
  }
             if       (IRcompare(numberpulses,       RotationalSignal,
sizeof(RotationalSignal) / 4)) {
    Serial.println("ROTATION");
    bitSet(leds, 1);
    updateShiftRegister();
    buzzerReaderState = digitalRead(buzzerReaderPin);
    if(buzzerReaderState == HIGH) {
      tone(buzzerPin, 260);
    } else {
      noTone(buzzerPin);
    }

    bitSet(leds, 2);
    updateShiftRegister();
    motorReaderState = digitalRead(motorReaderPin);
    if(motorReaderState == HIGH) {
      analogWrite(motorPin, 50);
    } else {
      analogWrite(motorPin, 0);
    }
    delay(500);
  }

  noTone(buzzerPin);
  analogWrite(motorPin, 0);
  delay(500);
}

void updateShiftRegister()
{
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}

//KGO: added size of compare sample. Only compare the minimum of the
two
boolean IRcompare(int numpulses, int Signal[], int refsize) {
  int count = min(numpulses, refsize);
  Serial.print("count set to: ");
  Serial.println(count);
  for (int i = 0; i < count - 1; i++) {
    int oncode = pulses[i][1] * RESOLUTION / 10;
    int offcode = pulses[i + 1][0] * RESOLUTION / 10;

#ifdef DEBUG
    Serial.print(oncode); // the ON signal we heard
    Serial.print(" - ");
    Serial.print(Signal[i * 2 + 0]); // the ON signal we want
#endif

    // check to make sure the error is less than FUZZINESS percent
     if ( abs(oncode - Signal[i * 2 + 0]) <= (Signal[i * 2 + 0] *
FUZZINESS / 100)) {
#ifdef DEBUG
      Serial.print(" (ok)");
#endif
    } else {
#ifdef DEBUG
      Serial.print(" (x)");
#endif
      // we didn't match perfectly, return a false match
      return false;
    }


#ifdef DEBUG
    Serial.print("  \t"); // tab
    Serial.print(offcode); // the OFF signal we heard
    Serial.print(" - ");
    Serial.print(Signal[i * 2 + 1]); // the OFF signal we want
#endif

     if ( abs(offcode - Signal[i * 2 + 1]) <= (Signal[i * 2 + 1] *
FUZZINESS / 100)) {
#ifdef DEBUG
      Serial.print(" (ok)");
#endif
    } else {
#ifdef DEBUG
      Serial.print(" (x)");
#endif
      // we didn't match perfectly, return a false match
      return false;
    }

#ifdef DEBUG
    Serial.println();
#endif
  }
  // Everything matched!
  return true;
}

int listenForIR(void) {
  currentpulse = 0;

  while (1) {
    uint16_t highpulse, lowpulse;  // temporary storage timing
    highpulse = lowpulse = 0; // start out with no pulse length

    //  while (digitalRead(IRpin)) { // this is too slow!
    while (IRpin_PIN & (1 << IRpin)) {
      // pin is still HIGH

      // count off another few microseconds
      highpulse++;
      delayMicroseconds(RESOLUTION);

      // If the pulse is too long, we 'timed out' - either nothing
      // was received or the code is finished, so print what
      // we've grabbed so far, and then reset

      // KGO: Added check for end of receive buffer
         if  (((highpulse  >=  MAXPULSE)  &&  (currentpulse  !=  0))  ||
currentpulse == NUMPULSES) {
        return currentpulse;
      }
    }
    // we didn't time out so lets stash the reading
    pulses[currentpulse][0] = highpulse;

    // same as above
    while (! (IRpin_PIN & _BV(IRpin))) {
      // pin is still LOW
      lowpulse++;
      delayMicroseconds(RESOLUTION);
      // KGO: Added check for end of receive buffer
         if  (((lowpulse  >=  MAXPULSE)   &&  (currentpulse  !=  0))  ||
currentpulse == NUMPULSES) {
        return currentpulse;
      }
    }
    pulses[currentpulse][1] = lowpulse;

    // we read one high-low pulse successfully, continue!
    currentpulse++;
  }
}
void printpulses(void) {
  Serial.println("\n\r\n\rReceived: \n\rOFF \tON");
  for (uint8_t i = 0; i < currentpulse; i++) {
    Serial.print(pulses[i][0] * RESOLUTION, DEC);
    Serial.print(" usec, ");
    Serial.print(pulses[i][1] * RESOLUTION, DEC);
    Serial.println(" usec");
  }

  // print it in a 'array' format
  Serial.println("int IRsignal[] = {");
  Serial.println("// ON, OFF (in 10's of microseconds)");
  for (uint8_t i = 0; i < currentpulse - 1; i++) {
    Serial.print("\t"); // tab
    Serial.print(pulses[i][1] * RESOLUTION / 10, DEC);
    Serial.print(", ");
    Serial.print(pulses[i + 1][0] * RESOLUTION / 10, DEC);
    Serial.println(",");
  }
  Serial.print("\t"); // tab
  Serial.print(pulses[currentpulse - 1][1] * RESOLUTION / 10, DEC);
  Serial.print(", 0};");
}
```

The input from the remote that I was able to capture in previous steps of this process.

Example of Output